International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 6, Issue 1, Feb 2016, 53-68

TRANS STELLAR
•Journal Publications •Research Consultancy

EVALUATION MODEL FOR ESTIMATING REUSABILITY PARAMETER USING METRICES

ASHWIN TOMAR[1] & V M THAKARE[2]

[1]*Siddhant Institute of Computer Application, under University of Pune, Maval, Pune, MAH, Pune, India*

[2]*P G Department of Computer Science, SGB Amravati University, Amravati, MAH, India*

ABSTRACT

The evaluation model is proposed which emphasis on use of reusable artifacts in development. The reusable artifacts can be used in many forms in various areas to speed up development and reduce the cost of project. A case study of website is considered which uses components as artifacts using rapid application development technology. The evaluation model is proposed, implemented and various reusability parameters like coupling between object, lack of cohesion of method, depth of inheritance, number of children, complexity of code, response for class are estimated and reusability of class is predicted.

KEYWORDS: Coupling Between Object (CBO), Lack of Cohesion of Method (LCOM), Depth of Inheritance (DIT), Number of Children (NOC), Response for Class (RFC), Mc Cabe's Cyclometric Complexity(MCC), Maintainability Index(MI)

Received: Dec 10, 2015; **Accepted:** Jan 13, 2016; **Published:** Jan 21, 2016; **Paper Id.:** IJCSEITRFEB20167

Original Article

INTRODUCTION

Software reuse is the process of creating systems from existing software rather than building them from scratch. Artifacts are used in more than one project for speeding the work. Reuse of requirements, code, modules, components, design, architecure, test plans, test cases can be done in form of artifacts. The benefits from reuse are reduction in cost, development time and increase in productivity. There are many issues involved with reuse libraries like certification of components, configuration management, library access, selection of artifacts (if there is no exact match).

Metrics provide basis for analyzing, measuring, comparing, evaluating all characteristics of product, process (quality assurance and reuse). The practice of applying software metrics to a software process and to a software product brings knowledge about the status of the process or product of software in regards to the goals to achieve. Metrics are used to control the process i.e expected and actual measurement of process can be done. An academic website is considered as case study which uses components for its development. The various parameters for measuring reusability are collected from literature. The evaluation of reusable parameters like WMC(weight metric class), Depth of Inheritance (DIT), RFC (Response For Class), LOC (lines of code), NOC(Number of Children), CBO (Coupling between objects), Coupling, Dependency is done.

LITERATURE SURVEY

In late sixties the concept of software reuse was introduced by Mcllroy. Primarily the focus was made on code reuse and so at an organization level the first formal program was established at the Raytheron Mission

Division. The output was the gain of 50% productivity. The defense department of USA saved millions of dollars annually by increasing reuse practices by 1%. Companies also gained from that reused quality and productivity as suggested by V.Mellarkod et.al[1].

Darg Mcllroy(1968-1969) first introduced the idea of systematic reuse as the planned development and widespread use of software components. Many software companies have achieved success in reuse programs. They are IBM, Hewlett - Packard, Hitachi. Hardware designers made good progress in their field during 1970. Software reuse is the process of creating software systems from existing software rather than building them from scratch. Different individuals have viewpoints that depend upon their responsibilities. Developer view is that software reuse is collection and use of available assets, from Consumer view reuse is positive goal of increasing productivity, as per Project manager reuse is seen as increasing quality as Reuse librarian reuse is seen as configuration management, arranging and categorizing it. Reuse code is often more appealing than automatic code generation.CBSE is emphasizing on building system by reusing high quality configurable software components. It reduces its development cost, higher reliability, better maintainability, decreases the complexity and time of development process.

Kung-kiu Lau et.al[2], classified component on basis of two groups (a) Object based - Javabeans, EJB, COM, .NET, CCM, Web Services, Fractal (b) Architecture based - Acme like ASLs, UML2.0, Cobra, SOFA, PECOS.A component model provides a standard way to develop and use components and is expressed by a set of conventions. These convention include “the standards structure of a component interfaces, the mechanism by which a component interacts with other components, patterns for asking a component about its feature and a means for browsing active components“.

M**.** Morisio et.al[3], C.Catal et.al[4], K. Sherif et.al[5], B. Jalendar et.al[6] investigated barriers of software reuse which are low investment, no specific process, lack of adaptation of process, no systematic approach, lack of support from management, lack of technical engineers interest, insufficient resources, low resource allocation, financial constraints i.e. high cost, lack of tool support, no repository, negativity effects, one dimensional solution, lack of specification for components, lack of certification of components, lack of review of requirements, lack of core competencies of organization, sufficient training, lack of tools and techniques.

Yong-liu et.al[7], B. Jalender et.al[8] worked on factors which facilitate reuse. These factor should be taken care of which increase reuse. They are reducing process risk, complying to standards, developing repositories, reuse organizational support, making reusability generic, certification of components, increasing commonality among application, tools support for retrieving components, solving problems associated with organization, management, process, assets, trust, culture, technology, architecture, lack of legal and contractual issues.

Govardhan et.al[8], categorized reuse by nature of environments in which software artifacts are reused. Vertical reuse occurs when software artifacts are reused in different projects in the same application domain. Horizontal reuse occurs when a software artifact is reused in a different application domain from the one in which it is originally developed. The reuse are of two kinds adhoc reuse and systematic reuse. The benefits of reuse are best achieved from a systematic software reuse approach. This involves the analysis, measurement and management of software systems and reuse libraries. The systematic reuse is a disciplined process of software development that makes use of existing software artifacts whenever they are available and their reuse is practiced. It is based on two related processes the design and development of reusable components and the utilization of reusable components.

L.A. Burgareli et.al[9] defined strategy for the generation of Platform Independent Model (PIM) for the Brazilian Satellite Launcher (BSL) software, based on the Model Driven Architecture (MDA) approach. To accomplish it, a generic software model is created, using design patterns, to represent a family of satellite launcher vehicles whose characteristics similar. From this generic model, PIMs are generated for variations of launcher vehicles, based on MDA approach. This strategy can contribute to the development of the software for same-family satellite launcher vehicles, as it defines a systematic for activities related to software modeling, and also allows the increase of BSL software models reuse.

Frakes and Terry et al[10], was first person(1996) to propose metric and models on software reuse. He suggested models based on cost benefits, assessing the maturity, the degree of reuse, the failure modes, and reuse library metrics. Models are given on the basis of reusability. Models are proposed on the basis of process, metrics, framework, neural network.

Gill & Tomar et.al[11], proposed a modified development process of CBSE on two approaches i.e composition based approach and generation based approach. CBSE promises many advantages, such as a shortened product development time, reductions in total costs, and fast access to new technology. Reused components would change more often than non-reused components because the reused components had to meet the requirements of percentage of modified code. The reuse needs software components to overcome managerial, legal, technical constraints.

Fazal-e.Amin et.al[12], proposed reusability assessment model which contains six attributes related to reusability of an SPL component: flexibility, maintainability, portability, scope coverage, understandability, variability. He suggested formulae's to find number of methods (NOM) in class, lines of code for components, coupling, cohesion, maintainability index, McCabe's Cyclometric Complexity (MCC), Flexibility, portability, depth of inheritance, scope coverage, reusability of class, coupling between objects, reusability of class.

Jiri Adamek et.al[13], presented his experience with cooperation in the area of component-based software engineering. There are a number of opportunities for cooperation. Each of these types of cooperation has its advantages and disadvantages. The big projects usually offer fine funding, provide co-operation among different types of partners (from big industry companies, SMEs, to universities and research institutes). On the other hand, management of these projects is quite complex and takes a significant amount of time.

Y. F. Chen et.al[14], Identifies four complexity issues for the engineering of large-scale software reuse: Acquisition, Classification, Representation, and Retrieval. A multimodeling framework is proposed. In this framework, each component is defined via four different generic models. All these models are incoporated onto a single reuse framework. The development of this proposed framework has been implemented in a proof of-concept reuse system prototype.

S. Kalaimagal et.al[15], emphasis on selection of use of high quality components, certification of components, evaluation of components using quality models. The components used lack the information from component vendors. Authors suggested component quality models with a minimum set of attributes that serves as a standard for the evaluation of software components. They are quality characteristic, attributes and metrics which are specific to the particular nature of components and CBSD. The estimation of components are based on objective, scope, metrics, methodology, description, metrices used and critiques.

Andreou, Tziakouris et.al[16], suggested reuse of software components which has the ability to significantly reduce both the software development costs and the development life cycle. Resultantly the time to market is also reduced because the complexities are resolved by the use of readymade components. In the CBSE the proper search and selection process of components is considered the corner stone for the development of any effective component based system.

N.Paliwal.et.al[17], mentioned about different types of artifacts like requirements, specification, architecture/design and its documents, templates, code, components, test plan. The information collected about artifacts to be reused are algorithm, plans classes, modules, code (source, complied), test cases, interfaces, schedules, strategies, data, experiences, documents (newly developed assets, negotiations,, vendors), patterns, components and many other things. These artifacts are used for evaluation of their quality and reusability parameter. It is likely that in some case during evaluation one attribute is not available then other can be used. The similar type of idea is shared by M. Goulao et.al in their work.

Miguel Goulao et.al[18] proposed quality model for component based development along with suitable metrics to assess components and assemblies. It can mitigates the shortcomings of models, particularly if both the models and its metrics are validated.

Koziolek et.al[19], proposed many approaches for evaluating of performance prediction and measurement of component based system based on software design. The main purpose of this analysis is to determine the maximum capacity, identify performance of critical components and to remove performance bottleneck. Factors influencing component performance are component implementation, required services, deployment platform, usage profile, resource contention.

D.C.Rine et.al[20], proposed reuse reference model to improve the competitive edge and time-to-market of many software development enterprises. The study of the a model is carried out using four steps. First, the reference model developed is based on the existing software reuse concepts. Second, the reference model is an empirical study which uses both legacy studies and lessons learned studies. Third, the impact of the reference model on software development effort, quality, and time-to-market is empirically derived. Fourth, an initial set of successful cases, which are based on the software reuse reference model utilization, are identified.

The proposed models suggested by William Frakes et.al[21], Wang Hong et.al[22], Mili et.al[23], Zhuo Kang[24], Fazal-e-Amin et.al[25], Xunmei et.al[26], kung-kiu Lau[27] are based on metric, pattern, reuse cost-benefits, maturity assessment, amount of reuse, failure modes, reusability metrics, reuse library metrics, reusable components.Some researchers have proposed models on reuse based on various techniques like Neuro-Fuzzy, Genetic algorithm.

Research paper explains about the benefits of reuse which are increase in production, quality, reliability, decrease in costs, reduction in time and intellectual energy. It emphasizes for development of reuse repository as a base for knowledge, minimizing the amount of development and work, risk for new projects. It increases efficiency of programmers, reduction in training, easy building of prototype and less time to marketing. Failure of software reuse is loss of precious time and money, loss of resources, loss of market share.

As per Dr Caper Jones in his book "The Economics of Software Quality" it is possible to reuse requirement, architecture, design, source code, test cases, test material, documents, database, portions of website. Reuse material can be harmful due to bugs and latent defect and hence needs to be certified to zero level of defects. Defect removal is by

inspection of code, document, reusable material, static analysis of code, formal testing of all code. The benefits of use of certified material are more in long run in terms of cost, duration of use. Defect prevention is less in case of bigger application. The use of pattern based design and development are example of reuse at higher level of abstraction. The libraries, API (application programming interface), framework, pattern, code all are reusable.

There are different approaches to reuse as described by Sommerville which are Design pattern, Program libraries, Special Product line (SPL), COTS Integration (Commercial Off Shelf), Program generator, Aspect Oriented Software development, Legacy System Wrapping, Configurable Vertical Applications, Service Oriented System, Application Frameworks, Component Based Development(CBSE) Domain Engineering, Reusable Components.

While searching research papers it was found that maximum papers which were collected are on Components(CBSE), Special product line(SPL) and Domain Engineering(DE). The software components can be Active X, VBX, COM objects, Java Beans, Window Services, CORBA Components. Software reuse is promoted through domain engineering, design pattern, product lines, frameworks, component based development.

A. Alvaro et.al[28] refers to software reusability that reuse potential of a software assets and module can be adapted in more than one software system. Based on data through online questionnaire results of software reusability approaches given by is visible among Malaysian IT industry are Design Patterns(63%), component based Development (45%), Application Frameworks COTS Integration(62%), Service-oriented Systems(56%), Application Product Line(37%), Legacy System Wrapping (24%), Program Libraries(7%), Program Generators(7%), Aspect-oriented Software Development (11%),Configurable Vertical Applications(7%).

REUSE MODEL

Frakes and Terry(1996): Was first person to propose metric and models on software reuse. He suggested models based on cost benefits, assessing the maturity, the degree of reuse, the failure modes, and reuse library metrics as suggested by N.S.Gill[29]. Models are proposed on basis of reusability, process, metrics, framework, neural network. The reuse can be of requirements stored in repositories, architecture design, database tables, code, test cases, patterns, components. Reuse of components are mainly in three areas component based software engineering(CBSE), special product line(SPL) & component off shelf (COTS).

Component models like COM and CORBA (i.e common object request broker architecture) allow software engineers to plug together components in different languages and platforms e.g spreadsheet, word processing, drawing and database application. Component is a computational unit and can be code, clients, servers, database, filters and layers in Hierarchical level. Component is defined as “a small binary object or program that performs a specific function and is designed in such a way to easily operate with other components and applications”. The component are developed using different technology i.e Object Management Group(OMG)-CORBA, Microsoft - COM, DCOM, SUN Microsystems – JAVABEANS.

There are various interaction**(Shaw & Garlan, 1996)** types among components like Procedure call, Data flow, Implicit Event Propagation, Message passing, Shared data, Instantiation, Knowledge passing. **CORBA i.e Common Object Request Broker Architecture** was proposed by Object Management Group to support open distributed communication between objects across a wide variety of platforms, languages. **JAVABEANS** is a reusable software components that can be manipulated visually in a builder tool. JAVA Beans depends on java languages and as java is

portable java beans are also portable. JAVA language provides interface for java beans. It supports interface inheritance, interface registration and gets automatically registered (like CORBA, COM) save developers time. EJB is an extension of java beans. EJB retains the basic java bean components system, but adds to it's the functionality of encapsulating a java bean component as a CORBA components.

MEASURING REUSABILITY PARAMETERS USING METRICS

It is to minimize the repetition of work, development time, cost efforts and increase the reliability of system. It is associated with Interfaces (has public methods), interface complexity(more complexity then more efforts required), customizability(changes as per customer), understandability (ease of use), portability (how easily the component are reused, transferred on new platforms),changeability (easily modifiable), documentation quality, coupling, inheritance, modularity, adaptability etc. There are some questions about reusing components described by Faizal-e-Amin et.al[30] which are as below:

- How much variability is there in the components? a) What is the average no of methods per class ? b) Number of methods/ Total no of classes c) What is the average no of children's per class?
- How it is easy to understand the components ? (a) What is the size of the component i.e Number of methods (NOM) Lines of code (LOC) (b) How much coupling is there in the components? i.e Coupling between the objects (CBO) c)How much cohesion is there in the components ? i.e Lack of cohesion in methods (LOCM) d) How much comment lines are there in components? i.e No of comments
- How easy is it to maintain the system? (a) Maintainability index (MI) - It is a software metric which measures how maintainability i.e. easy to support and change the source code is. (b) McCabe's Cyclometric Complexity (MCC) - it is used to measure the complexity of code.
- How much flexibility is there in the components? a) How much coupling is there in the components
- (b) How much cohesion is there in the components
- How portable is the components? a)How independent is the component i.e Depth of Inheritance (DIT)
- How much of the scope is covered by the component ? How many features are covered by the component i.e NOM / Total number of methods in all classes.

Reusability Metrices

Reusability metrices described by Faizal-e-Amin[30] to measure reusability with formulae are

- **Maintainability** = (*0.5 * adjusted MCC) + (0.5 * adjusted MI)]*
- **Portability** = Independence = 1- adjusted DIT
- **Flexibiity**(F) = 1- [(0.5 * coupling) + (0.5 * Cohesion)]
- **Understandability** (U) - *U = 1- [(0.25 * Coupling) + (0.25 * Cohesion) + (0.25 * Comments) + (0.25 * Size)*
- **Scope coverage** = NOM / Total number of methods in all classes
- **Size Metrics** = [(*0.5 *adjusted LOC) + (0.5 * adjusted NOM)]*

- **Coupling =** adjusted CBO
- **Cohesion =** adjusted LCOM
- **Variability metric** = 0.5 * (NOC / Total no of classes) + 0 .5 * (NOM / total no of methods in all classes)
- **Reusability of class** = 0.16 * flexibility + 0.16 * understandability + 0.16 * Portability + 0.16 * Scope coverage + 0.16 * Maintainability + 0.16 * Variability

Reuse and reusability are two major aspects in object oriented software which can be measured from inheritance hierarchy. Metrics used to evaluate reuse and reusability of object oriented software are Depth of Inheritance Tree(DIT), Number of Children(NOC), Method Inheritance Factor(MIF) and Attribute Inheritance Factor (AIF). ***Reusability is a property of a software asset that indicates it's probability of reuse***. The complexity of a method is evaluated as follows.(McCall Cyclometric complexity).

C[Mj] = Number of edges – Number of vertices + 1 (in a particular flow graph) or C[Mj] = Number of Decision Elements of a method + 1

Reusability is the core principle of Service Oriented Architecture (SOA). Reusability of service is the degree to which the service can be used in more than one business process or service applications without having much to configure and invoke it. Benefits of Service reuse is that i) reuse reduces the development time and cost, ii) reuse minimizes the duplication of services iii) makes maintenance easier, iv) reusing of service increases the productivity thereby increase return On Investment v) the reliability of the service increases and overall risk reduces. vi) reusing a new business process can be assembled from the existing services to meet the current consumer's requirements and market needs[31].

PROPOSED EVALUATION MODEL

While developing website different reusable artifacts are used. The artifacts can be used in different phases to speed up the development and reduce cost. The left side of Fig 1 shows reusable artifacts as Input and right side shows estimation of reusability parameters like NOC, DIT. Here components are used as reusable components. Since components are used while developing, their reusability parameters is estimated. The reusability is evaluated by testing code of using metrics in form of Coupling, Cohesion, Depth of Inheritance, Coupling between Objects, WMC, RFC. The reusability is evaluated by testing code, components using metrices in the form of Coupling, Cohesion, Depth of inheritance, Coupling between object, WMC, RFC.

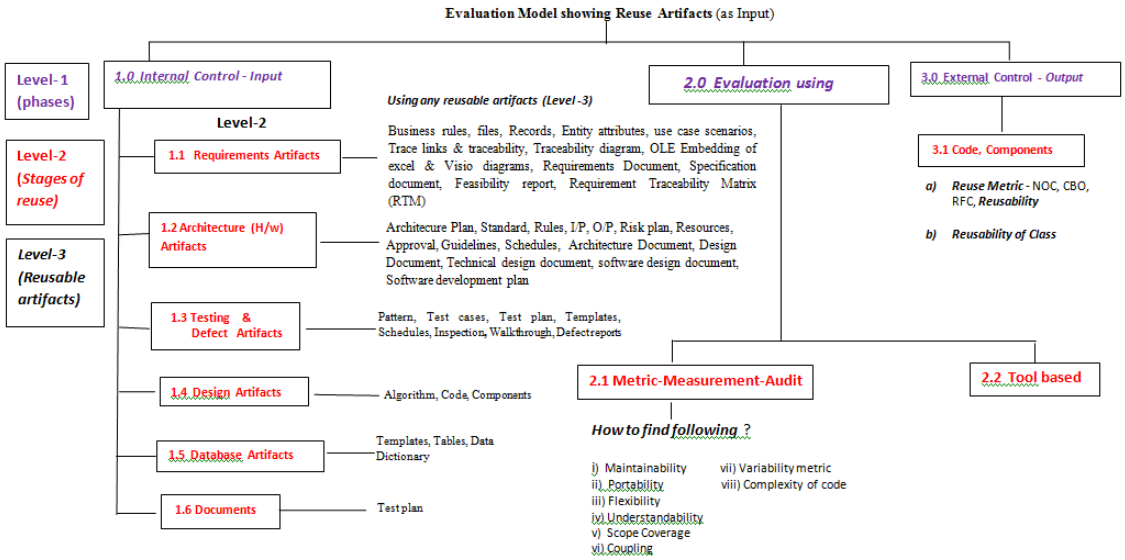

Figure 1: Shows Evaluation Model for Estimating Reusable Parameter (as Output) using Metrics

CASE STUDY OF WEBSITE SHOWING REUSE OF COMPONENT

An academic website developed by faculties for Siddhant Institute of Computer Application under url: www.siddhantica.edu.in **is considered as case study.** Presently the developed website is replaced by new website under same url as owner sold the college so the new owner hosted a new website under the same url. Evaluation work started earlier, as website was removed and so the work continued considering it as sample project website. It was developed using reusable artifacts components as shown in Fig 1 Evaluation model for estimating reusability parameters using metrics. The information about website development and others is **Platform** used – Windows , **Database** - MySQL, **Server** - Net4, **Languages used** – HTML, Javascript, XML, J Query, **Lines of Code** -10000, **Person involved** – 5, **Duration** –3 months (during vacation) ,**Hours/ per day** – 10 hrs, **Person who can login were** - Students, Visitor, Admin, Staff, **The Admin** - had read and write permission and others have read permission only. The steps involved in development of website are a) Initial design decided b) Color was selected c) Text combination with font size and type d) Standard form size, image, image size was selected. e) Planning was done for assuring quality, reuse (component, code, template, table), requirements, architecture designing, test plan, test cases. Installation of different software was done for developing the website which includes Dreamweaver 8, Adobe Photoshop 7.0, Max 5, Flash Factory, DHTML Manual, Visual Studio, SQL server 2008 R2, Crystal Report, MS Office , Window 7(64 bit).

The website has information on MCA, PGDM Courses, Faculty, Admission, Training & Placement, Facility (Library, Hostel, Transport). There are photos on library, hostel, transport as seen. The under "**VIEW**" functionality like *Institute*, *Campus*, *Apply Online for* (Admission enquiry, faculty recruitment), *Downloads* (Admission form, Mandatory disclosure, Information brochure), *Photo Gallery user* can be seen by just one click. The ***"Apply online"*** uses components by receiving information of students and staff during admission and recruitment vice versa. The code of component is considered for evaluating the reusability aspects of website. The website was developed using reusable component. Website can be big, small with variable number of reusable components. e.g Login forms - represents component A, Client represents component B, server - is component C. Different types of website are represented as .gov=government, org=non-profit organization, .com=commercial, business, edu=educational. Website uses certified reusable components

like Login, Tables, Report. The login form is already certified by Microsoft. It is being used while making the website using ASP.Net. The other component used are Student and Faculty information access code. The Reusable Components are Login, Table, Report, Student & Faculty code.

Greedy algorithm, AHP technique are used for optimal choice and decision making while evaluating the parameters for Case study. **A greedy algorithm** follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. While developing the website the activities were controlled like requirements, defects were removed and website was tested. Reuse is associated with CBO, NOC, DIT, WMC, RFC, NOM.

Figure: 2 Academic Website

The crystal report is component of visual studio for making report. The report is on the number of students enquired for admission, no of enquiry received college wise, no of post applied by faculties dept wise. The rrequired ffunctionality for System are a) Add students b) Remove students c) Add students details d) Add faculty e) Remove Faculty f) Add faculty details. The functionalities provided by the components are a) Get students b) Create students c) Generate a number for students d) Get faculty e) Create faculty f) Generate a number for faculty. The Operation involved in the Components are a) Get students() b) Get faculty() c) Get details of faculty().

Table 2: Name of the Components with No of classes

Sr. No	No	Component Name	Usefulness	Classes No	LOC
1	A	Staff Component	Code	5	50 -150
2	B	Surdents Component	Code	7	100
3	C	Login Form - Microsoft certified	Security	5	500
4	D	Report as Component	Reporting	-	-

Evaluation of Reusability Parameters

- **Mc Cabe Cyclometric Complexity**

Cyclometric Complexity testing is a process that measures the complexity of a program. The steps to calculate the complexity is: (i) A program structure is represented through flow chart i.e pictorial way of representing. Step (ii) Flow chart is converted into a control flow graph containing only nodes and edges. The steps to convert into flow graph are listed down: a) Identify the predicates (decision points) in flow chart b) Ensure predicates are simple else break up a condition into simple predicates.

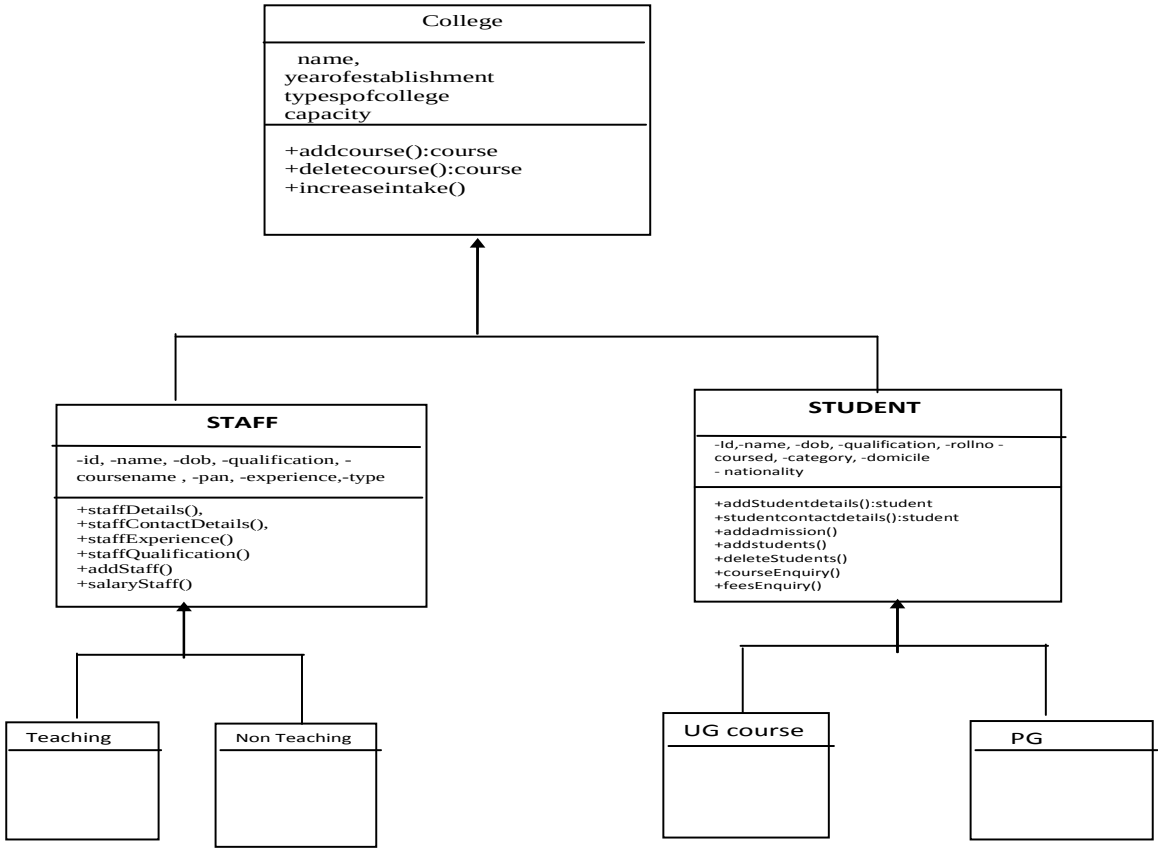

Figure 3: Showing Classes with Attributes and Function

c) Combine all the sequential statements into a single node. d) If a set of sequential statement is followed by a simple predicates, combine all the sequential statements and the predicate check into one node. Such nodes with two edges emanating from them are called predicate node. e) Make sure that all the edges terminate at some same node. Add an extra node at the end to represent that all the set of sequential statements reach to that node. Step (iii) Calculate Cyclometric Complexity (MCC) using one of either two methods:

The Complexity of Method is Evaluated as

- From flow chart, MCC = no of decision elements of a method (predicates) + 1 or
- From flow graph = No of edges – No of vertices (nodes) + 1

The evaluation of Student components is performed. The component comprises ***Student.Enquiry***, ***Student.Recruitment, Student.Report.*** The Feedback pat is common among student and staff components. In short the website has Enquiry form, Recruitment, Feedback form which is integrated into single component. The degree of reuse

of component is evaluated. using Complexity(MCC) = no of decision points in a method + 1 i.e MCC

(for Staff.recruitment) = 1 + 1 = 2

- **Coupling / Dependency:**

Coupling signifies the level to which one module depend on the other module. It is using the functionality of the other module. Coupling helps in determining the quality of design in software.

Table: 3 Shows Values of Various Reusability Parameters of Student Component

Components	**NOC**	**NOM**	**CBO** Coupling	**DIT**	**WMC**	**RFC (NOM +1)**	**MCC**	**MI**	**Comment Lines**	**LCOM** Cohésion	**Size = NOM /LOC** LOC is lines of **Code**	
Staff. Report	1	6	2	4		6	1	15	6	1	6/13 = 0.461	STAFF COMP ONENT
Staff. Recruitment	1	7	0	4		0	9	15	0	1	7/13 = 0.53	
Total	2	13										
Student.enquiry (admission)	1	4	**2**	2	5	6	3	15	5	1	4/250	STUDE NT COMP ONENT
Student.report (by college)	1	1	1	2	3	2	3	15	1	1	1 /100 = 0.01	
Total	4	10	4									
Feedback (Common component)	1	2	1	3	0	3	0	15	4	1	2/40	FEEDB ACK COMP ONENT

- **Cohésion**

It measure how well the methods of a class are related to each other. A cohesive class performs one function. Lack of cohesion means that a class performs more than one function. In such a case it should be split. High cohesion means high encapsulation. Lack of cohesion of methods metrics aims to detect problem classes. A high LCOM means low cohesion. LCOM4 is for visual basic program and measures the number of connected components in a class. The various values of visual basic program is LCOM4 = 1 indicates cohesive class. LCOM4 >=2 indicates problem. The class should be split into smaller classes. LCOM4 = 0 indicates bad class as no method is in class. Student and Staff component code is not easy nor difficult to maintain and hence the value assumed is 0.5.

- **CBO (Coupling between objects**)

It is a count of the number of other classes to a class is coupled.

- **NOC (Number of Children)**

It is the measure that counts the children of a class. NOC itself shows the reuse of a class. A large number of children mean that the functionality of the class is reused through inheritance.

- **Depth of Inheritance (DIT)**

It is the length from the node where the class is located to the root of the tree. The depth of inheritance tree measures in the maximum length from the node to the root. It is the measure that indicates the depth of a class within a hierarchy.

- **RFC (Response For Class)**

(i) One measure of Coupling is RFC. This measures the complexity of the class in terms of method calls. *It is calculated by adding the number of methods in the class* **(not including inherited methods)** *plus the number of distinct method calls made by the methods in the class* **(each method call is counted only once even if it is called from different methods). The** *RFC is the "Number of Distinct Methods and Constructors invoked by a Class.*" **The response set of a**

class is the set of all methods and constructors that can be invoked as a result of a message sent to an object of the class. This set includes the methods in the class, inheritance hierarchy, and methods that can be invoked on other objects. ii) *If the RFC for a class is large, it means that there is a high com***plexity.**

- **LOC (lines of Code)**

It is the measure of the lines of source code. It is the size indicator of the entity. The size of component depends on number of method divided by lines of code.

- **Maintainability Index**

It is a software metric which measures how maintainable the source code is. The MI is calculated as a factored formula consisting of Lines of code(LOC), Cyclomatic Complexity(G), Halstead volume (V) and percent of lines of comment (CM). Maintainability Index has been re-set to lie between 0 and 100. As the code tendered towards 0 it was hard to maintain code. The higher the value of MI more easy (code) is to maintain.

MI = MAX(0, 171 - 5.2 * In (Halstead Volume) – 0.23 * MCC – 16.2 * ln(LOC)) *100/171)

Visual Studio provided an interpretation i.e MI >-20 is High Maintainability, 10 <= MI < 20 indicates Moderate Maintainability and MI < 10 is for Low Maintainability. Since the code is easy to maintainable so MI is considered as 15 as the value of Hastead volume is not there.

Finding Reusability of Student.Enquiry (Admission Enquiry by Students)

Reusability of class = 0.16 * flexibility + 0.16 * understandability + 0.16 * Portability + 0.16 * Scope coverage + 0.16 * Maintainability + 0.16 * Variability

- **Flexibility**(F) = 1- [(0.5 * coupling) + (0.5 * Cohesion)]
- Flexibility of Student.Enquiry = 1-[(0.5) * 2) + (0.5 * 1)] = **0.5 x 0.16 = - 0.08**
- **Understandability** (U) = *1- [(0.25 * Coupling) + (0.25 * Cohesion) + (0.25 * Comments) + (0.25 * Size)*
- Understandability of Student.enquiry = 1-[(0.25 * 2) + (0.25 * 1) + (0.25 * 5) +
- (0.25 * 4 / 250) = 1-2.004 = 1.004 **x 0.16 = - 0.16064**
- **Portability** = Independence = 1- adjusted DIT
- Portability of student.enquiry = 1- (2) = -1 = **-1**
- **Scope coverage** = NOM / Total number of methods in all classes
- Scope coverage of student.enquiry = 4 /10 = **0.4**
- **Maintainability** = (*0.5 * adjusted MCC) + (0.5 * adjusted MI)]*
- Maintainability of student.enquiry = (0.5 * 3) + (0.5 * 15) =1.5 + 0.75 = 2.25
- **Variability metric** = 0.5 * (NOC / Total no of classes) + 0.5 * (NOM / total no of methods in all classes)= 0.5 * (1 / 4) + 0.5 * (4/10) = **0.062 + 0.20 = 0.2625 x 0.16 = 0.042**

The calculated value for student.enquiry is used to find reusability of class.

Reusability of class = 0.16 * flexibility + 0.16 * understandability + 0.16 * Portability + 0.16 * Scope coverage + 0.16 * Maintainability + 0.16 * Variability = - 0.08 - 0.16064 -1 + 0.4 + 2.25 + 0.042 = 1.45136

Reusability of Class (Student.enquiry) is found to be = ***1.45136***

Similarly the reusability of student.recruitment, student.report and hence Staff component can be found .

Finding Reusabillty of - Student.Report

Reusability of class = 0.16 * flexibility + 0.16 * understandability + 0.16 * Portability + 0.16 * Scope coverage + 0.16 * Maintainability + 0.16 * Variability

- **Flexibility**(F) = 1- [(0.5 * coupling) + (0.5 * Cohesion)]
- Flexibility of Student.Report = 1-[(0.5) * 1) + (0.5 * 1)] = 0 x 0.16 = 0
- **Understandability** (U) = *1- [(0.25 * Coupling) + (0.25 * Cohesion) + (0.25 * Comments) + (0.25 * Size)*
- Understandability of Student. Report = 1-[(0.25 * 1) + (0.25 * 1) + (0.25 * 1) + (0.25 * 1 /100)
- = 0.25 + 0.25 + 0.25 + 0.0025 = -0.0396
- **Portability** = Independence = 1- adjusted DIT
- Portability of student. Report = 1- (2) = -1 x **0.16** = **-0.16**
- **Scope coverage** = NOM / Total number of methods in all classes
- Scope coverage of student. Report = 1 /10 x **0.16** = **0.016**
- **Maintainability** = (*0.5 * adjusted MCC) + (0.5 * adjusted MI)]*
- Maintainability of student. Report = (0.5 * 3) + (0.5 * 15) = **9 x 0.16** = **1.44**
- **Variability metric** = 0.5 * (NOC / Total no of classes) + 0.5 * (NOM / total no of methods in all classes) = 0.5 * (1 / 4) + 0.5 * (1/10) = (0.125 + 0.05) x **0.16** = **0.028**

The calculated value for student. Report is used to find reusability of class.

Reusability of class = 0.16 * flexibility + 0.16 * understandability + 0.16 * Portability + 0.16 * Scope coverage + 0.16 * Maintainability + 0.16 * Variability

= 0 – 0.0396 – 0.16 + 0.016 + 1.44 + 0.028 = 1.2844

Reusability of Class Student. **Report** is calculated to be = **1.2844**

Similarly the reusability of Feedback & Staff component can be found.Therefore the degree of reuse of component STUDENT is sum of module **Student. Enquiry** and **Student. Report**

Reusability of Component –STUDENT - = Student. Enquiry + Student.Report = **1.45136 +** 1.2844 **= 2.73**

Average of two Components is Found to be = 1.36788

Similary Reusability of Component Feedback is calculated as = **1.426**

Reusability of Component Staff.Report is calculated as = **0.8119**

Reusability of Component Staff .Recruitment is calculated as **= 1.7491**

Therefore Sum (Staff Repot + Staff Recruitment) = 0.8119 **+** 1.7491 = **2.561**

RESULTS

The values of various reusability parameters are found. The reusability of student.enquiry module, student.report module, feedback module is estimated to be *1.451, 1.28, 1.426* respectively. Similarly the reusability of staff (component) i.e Report and Recruitment is found to be 0.8119, 1.7491 respectively. Therefore the values of reusability parameters for Student (Enquiry + Report), Student(feedback) and Staff(component) is found to be *2.73, 1.426, 2.561* respectively.

CONCLUTIONS

The above method can be used to find reusability of various components like COTS, SPL and many others. The values of reusability of components tell us which is the better than other. The reusability parameters can be ranked using AHP technique.

REFERENCES

1. *V. Mellarkod, R. Appan, D.Jones, K.Sherif," A multilevel analysis of factors affecting software developers intention to reuse software assets: An empirical investigation", Information and Management, Vol 44, 613- 625,2007.*
2. *kung-kiu Lau, Zheng Wang, "Software component models", IEEE Transaction of Software Engineering, Vol 33, No 10, Oct 2007.*
3. *M. Morisio, M. Ezran, C. Tully, "Success and Failure Factors in Software Reuse", IEEE Transactions On Software Engineering, Vol. 28, No.4, April 2002.*
4. *Cagatay Catal, "Barriers to the Adoption of Software Product Line Engineering", ACM SIGSOFT Software Engineering Notes, Vol 34, No-6, Nov 2009.*
5. *Karma Sherif, Ajay Vinze, "Barriers to adoption of software reuse a qualitative Study", Information & Management, Vol 41, No 2, pp.159-175, Dec 2003, www.elsevier.com.*
6. *B.Jalender, N.Gowtham, K.Praveen Kumar, K. Murahari, K. Sampath, "Technical Impediments to Software Reuse", International Journal of Engineering Science and Technology,Vol2 (11), 6136-6139, 2010.*
7. *Yong-liu, Aiguang-yang, "Research and Application of Software Reuse", Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, IEEE, Vol 12, pp. 1-5, 2007.*
8. *B. Jalender, Dr. A. Govardhan, Dr. P.Premchand, "A Pragmatic Approach to Software Reuse", Journal of Theoretical and Applied Information Technology, Vol 14, No. 2, pp 80-85, 2005-2010, www.jatit.org.*
9. *Luciana Akemi Burgareli, Selma, S.S. Melnikoff, Mauricio G. V. Ferreira, "A Software Model Reuse Strategy for Brazilian Satellite Launcher", 19th Australian Conference on Software Engineering, pp. 627-632, 26-28 March 2008.*
10. *William Frakes, Carol Terry, "Software Reuse: metrics and models", ACM Computing Surveys, pp. 415-435, Vol 28, No 2, June 1996.*

11. *Nasib Singh Gill, Pradeep Tomar, "Modified Development Process of Components-Based Software Engineering", ACM SIGSOFT Software Engineering Notes, Vol-35, No-2, March 2010.*
12. *Fazel-e-Amin, A.K. Mahmood, Alan Oxley, "Reusability Assessment of open Source Components for Software Product Lines", International Journal on New Computer Architectures and Their Applications (IJNCAA), The Society of Digital Information and Wireless Communication ,Vol 1, No 3, pp. 519-533, 2011.*
13. *Jiri Adamek, Petr Hnetynka, "Perspectives in Component based Software Engineering", ACM, SEESE, 13 May, 2008, www.eclipse.org.*
14. *Yufeng F. chen, W. Gerry Howe and Nazir A. Warsi , "A Multimodelling Framework for Complex Software Reuse", Proceedings of International Conference and Workshop on Engineering of Computer-Based Systems, pp. 81- 87, 1997.*
15. *S.Kalaimagal and R. Sinivasan, "A retrospective on software component quality models," ACM SIGSOFT Software Engineering Notes, vol. 3 no. 6, p. 1, Oct. 2008*
16. *A. S. Andreou and M. Tziakouris, "A quality framework for developing and evaluating original software components," Information and Software Technology, vol. 49, no. 2, pp. 122–141, Feb. 2007.*
17. *Nagesh Paliwal, Vivek Shrivastava, Ketki Tiwari, "An Approach to find Reusability of Software using object Oriented Metrics", Vol. 3, Issue 3, March 2014.*
18. *Miguel Goulao, "Component Based Software Engineering", OOPSLA ACM, October 16-20, 2005.*
19. *H. Koziolek, "Performance evaluation of component-based software systems a survey," Performance Evolution, vol. 67, no. 8, pp. 634–658, Aug. 2010.*
20. *D.C Rine, N Nada, "An empirical study of a software reuse reference model", Information and Software Technology, Volume 42, Issue 1, 1 January 2000, Pages 47–65.*
21. *William Frakes, Carol Terry, "Software Reuse: metrics and models", ACM Computing Surveys, pp. 415-435, Vol 28, No 2, June 1996.*
22. *Wang Hong, "Architecture-Centric Software Process For Pattern Based Software Reuse", World Congress on Software Engineering, Vol 4, pp. 95-99, 2009, IEEE.*
23. *A. Mili, S.Fowler Chmiel, R. Gottumukkala, L. Zhang, "An Integrated Cost Model for Software Reuse", Proceedings of the International Conference on Software Engineering, ACM, pp. 157-160, 4-11 June 2000.*
24. *Zhuo Kang, Yan Li, Li-shan Kartg, "Automatic Programming Methodology for Program Reuse", International Conference on Computational Intelligence and Security, Vol 1, pp 208-214, Nov 2006, IEEE.*
25. *Fazel-e-Amin, Ahmad Kamil Mahmood, Alan Oxley, "A Review of Software Component Reusability Assessment Approaches", Research Journal of Information Technology, Vol 3, No1, 2011.*
26. *Xunmei GU, Jun SHI, "Reuse Metrics for Object-Oriented Method", NCIPET-2012, Proceedings published by International Journal of Computer Applications (IJCA), Vol 2, No 11, pp. 6136-6139, 2010.*
27. *Kung-Kiu Lau, Zheng Wang, "Software Component Models", Transactions On Software Engineering, Vol. 39, No.10, October 2007.*
28. *A. Alvaro, E. Santana de Almeida, and S. Romero de Lemos Meira, "A software component quality framework," ACM SIGSOFT Software Engineering Notes, vol. 35, no. 1, p. 1, Jan. 2010.*
29. *N. S. Gill, "Inheritance Hierarchy Based Reuse & Reusability Metrics in OOSD," IJCSE, Vol. 3, no. 6, pp. 2300–2309, 2011.*

30. *Fazel-e-Amin, Ahmad Kamil Mahmood, Alan Oxley, "A Review of Software Component Reusability Assessment Approaches", Research Journal of Information Technology, Vol 3, No1, 2011.*

31. *T. Karthikeyan, G.Geetha, "A Study and Critical Survey on Service Reusability Metrics", I.J. Information Technology and Computer Science, MECS, Vol 5, pp 25-31, 2012.*

32. *O.Korkmaz, "System simulation for Software Quality Assurance," August, 2007, Atilim University, MS thesis.*